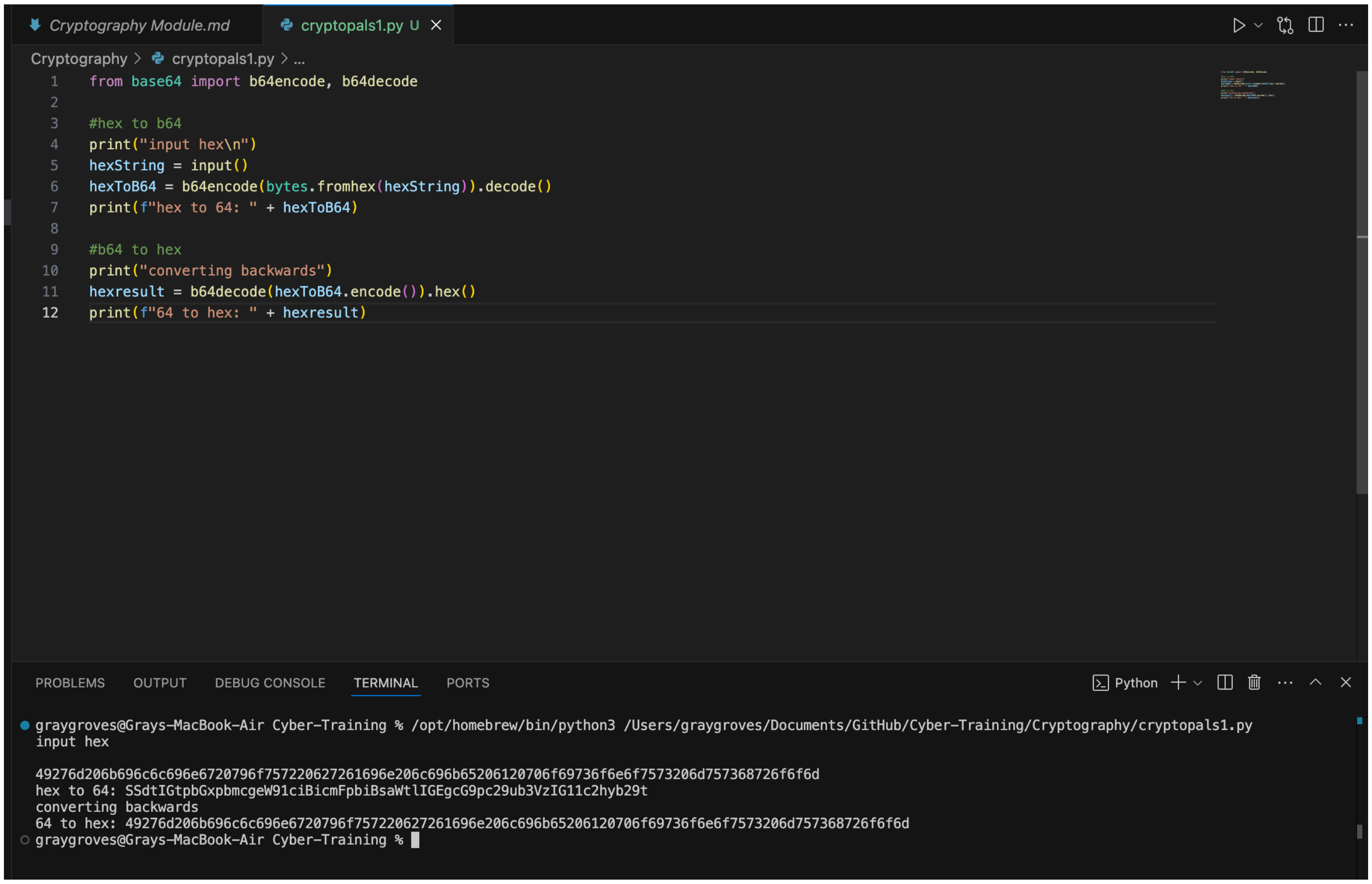
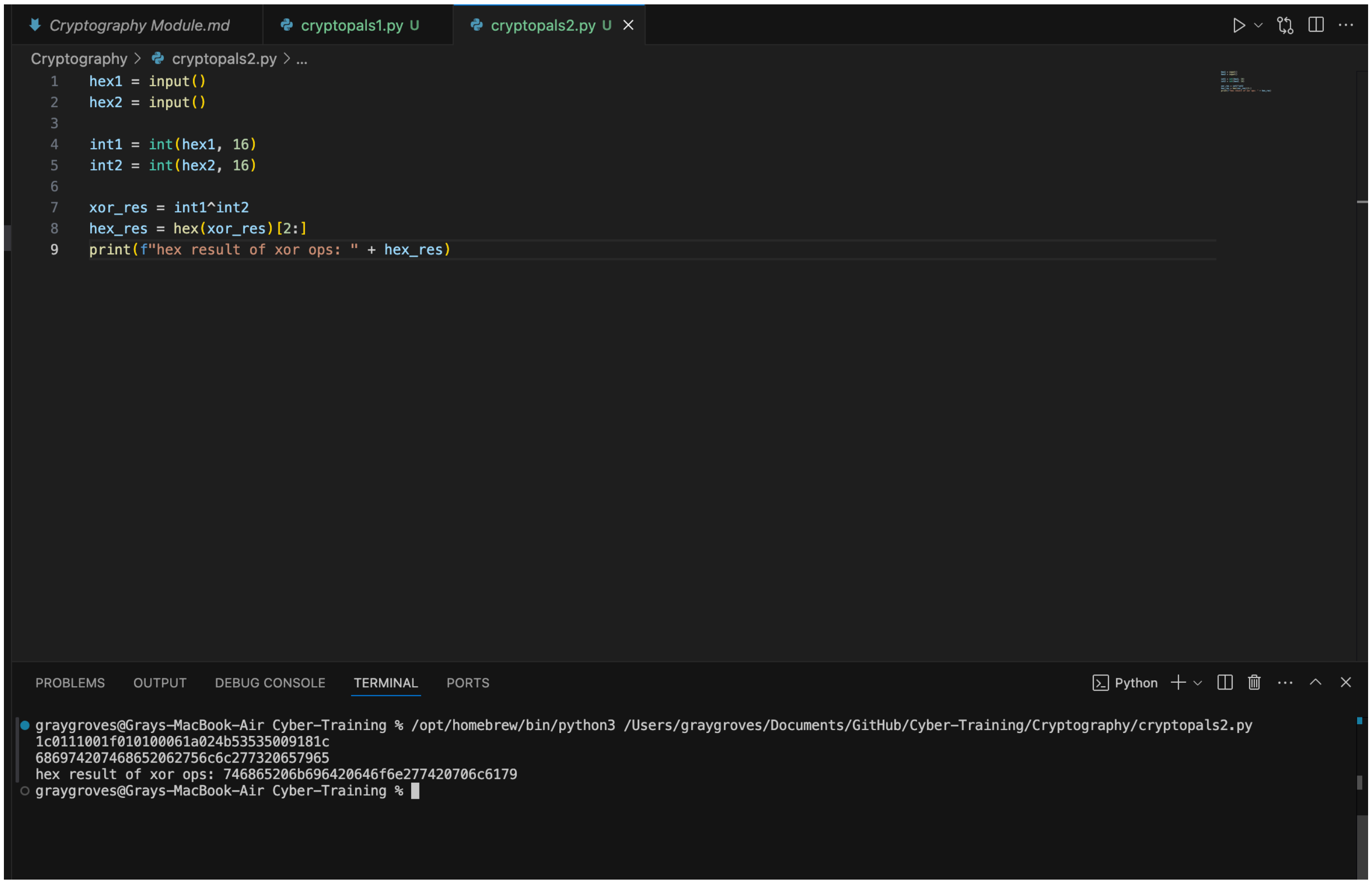
1:



```python
from base64 import b64encode, b64decode

#hex to b64
print("input hex\n")
hexString = input()
hexToB64 = b64encode(bytes.fromhex(hexString)).decode()
print(f"hex to 64: " + hexToB64)

#b64 to hex
print("converting backwards")
hexresult = b64decode(hexToB64.encode()).hex()
print(f"64 to hex: " + hexresult)
```

```
graygroves@Grays-MacBook-Air Cyber-Training % /opt/homebrew/bin/python3 /Users/graygroves/Documents/GitHub/Cyber-Training/Cryptography/cryptopals1.py
input hex

49276d206b696c6c696e6720796f757220627261696e206c696b65206120706f69736f6e6f7573206d757368726f6f6d
hex to 64: SSdtIGtpbGxpbmcgeW91ciBicmFpbiBsaWtlIGEgcG9pc29ub3VzIG11c2hyb29t
converting backwards
64 to hex: 49276d206b696c6c696e6720796f757220627261696e206c696b65206120706f69736f6e6f7573206d757368726f6f6d
graygroves@Grays-MacBook-Air Cyber-Training % 
```

2:


Cryptography Module.md | cryptopals1.py U | cryptopals2.py U

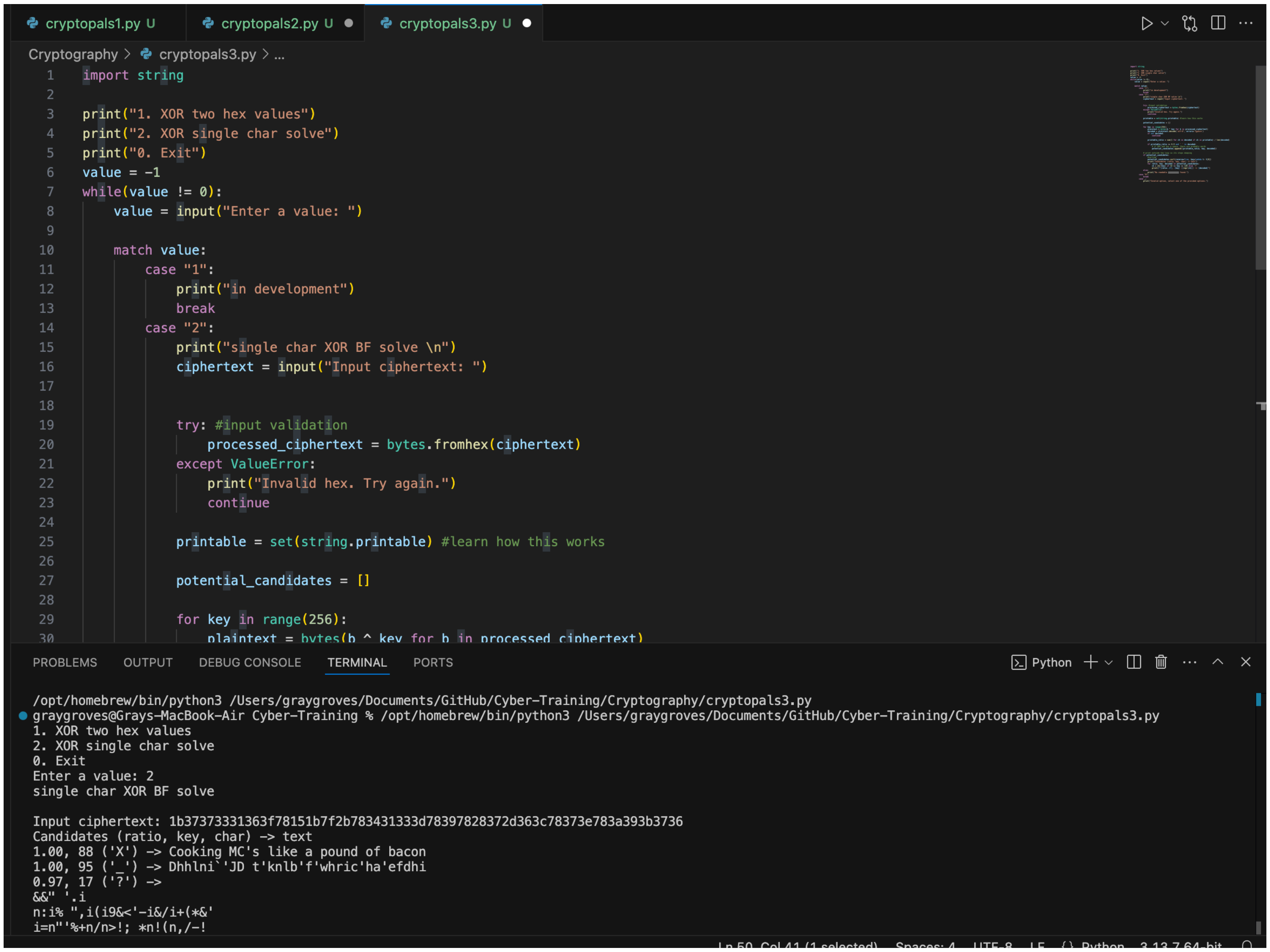
Cryptography > cryptopals2.py > ...

```python
hex1 = input()
hex2 = input()

int1 = int(hex1, 16)
int2 = int(hex2, 16)

xor_res = int1^int2
hex_res = hex(xor_res)[2:]
print(f"hex result of xor ops: " + hex_res)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
graygroves@Grays-MacBook-Air Cyber-Training % /opt/homebrew/bin/python3 /Users/graygroves/Documents/GitHub/Cyber-Training/Cryptography/cryptopals2.py
1c0111001f010100061a024b53535009181c
686974207468652062756c6c277320657965
hex result of xor ops: 746865206b696420646f6e277420706c6179
graygroves@Grays-MacBook-Air Cyber-Training %
```

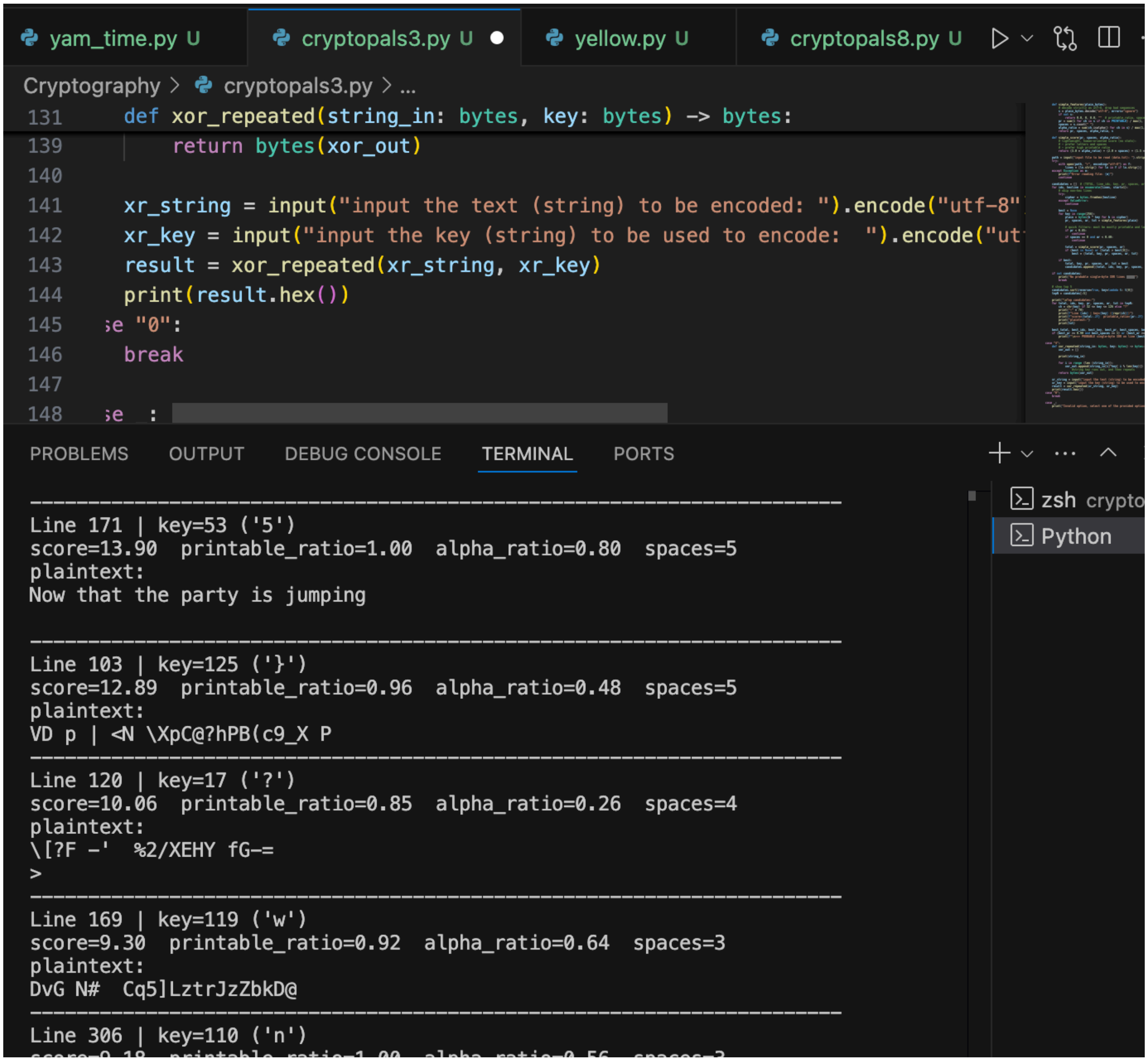
3: Key 88: Cooking MC's like a pound of bacon



```python
import string

print("1. XOR two hex values")
print("2. XOR single char solve")
print("0. Exit")
value = -1
while(value != 0):
    value = input("Enter a value: ")

    match value:
        case "1":
            print("in development")
            break
        case "2":
            print("single char XOR BF solve \n")
            ciphertext = input("Input ciphertext: ")


            try: #input validation
                processed_ciphertext = bytes.fromhex(ciphertext)
            except ValueError:
                print("Invalid hex. Try again.")
                continue

            printable = set(string.printable) #learn how this works

            potential_candidates = []

            for key in range(256):
                plaintext = bytes(b ^ key for b in processed_ciphertext)
```

```
/opt/homebrew/bin/python3 /Users/graygroves/Documents/GitHub/Cyber-Training/Cryptography/cryptopals3.py
graygroves@Grays-MacBook-Air Cyber-Training % /opt/homebrew/bin/python3 /Users/graygroves/Documents/GitHub/Cyber-Training/Cryptography/cryptopals3.py
1. XOR two hex values
2. XOR single char solve
0. Exit
Enter a value: 2
single char XOR BF solve

Input ciphertext: 1b37373331363f78151b7f2b783431333d78397828372d363c78373e783a393b3736
Candidates (ratio, key, char) -> text
1.00, 88 ('X') -> Cooking MC's like a pound of bacon
1.00, 95 ('_') -> Dhhlni``'JD t'knlb'f'whric'ha'efdhi
0.97, 17 ('?') -> 
&&" '.i
n:i% ",i(i9&<'-i&/i+(*&'
i=n"'%+n/n>!; *n!(n,/-!
```

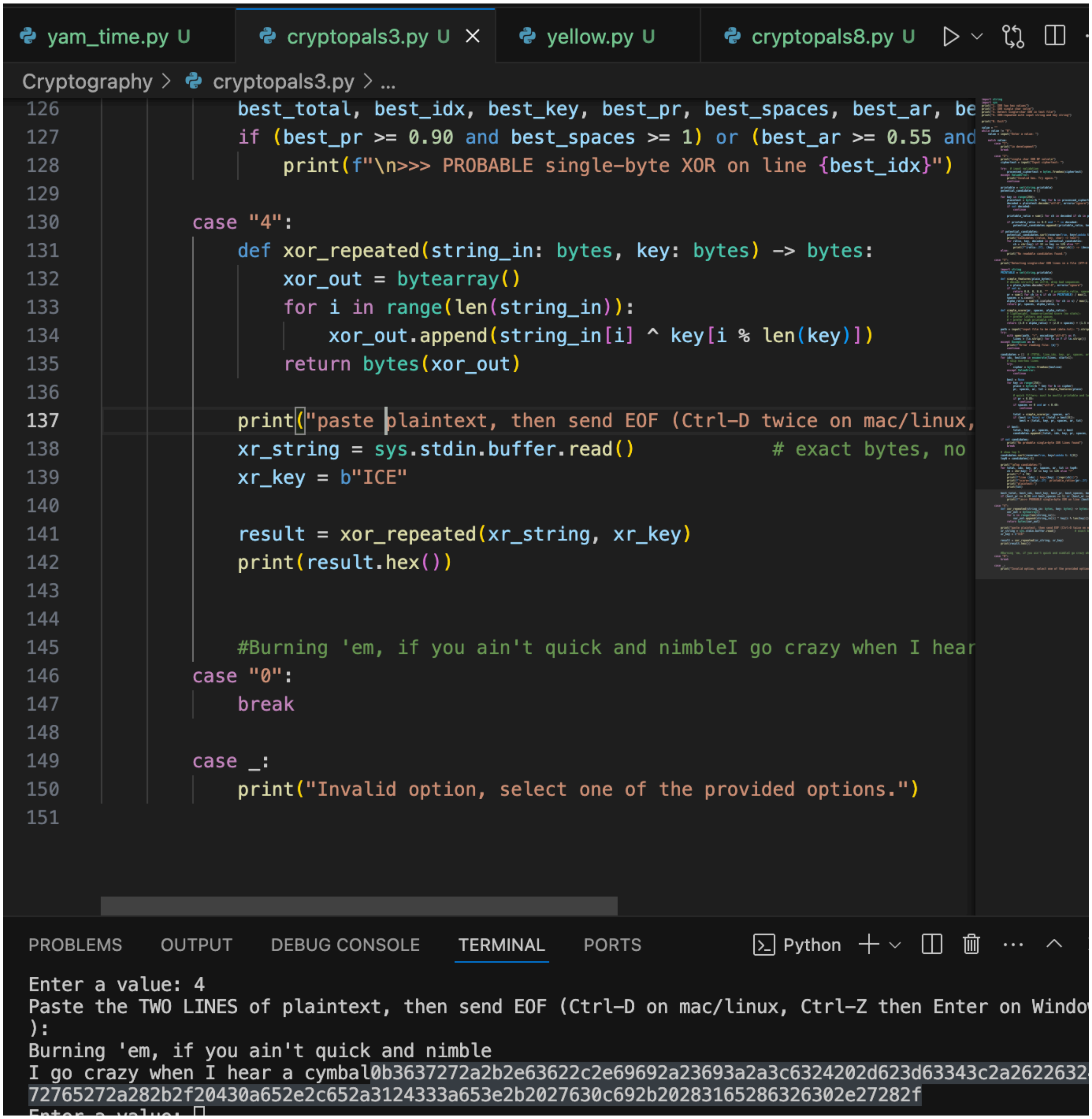
4:



```python
    def xor_repeated(string_in: bytes, key: bytes) -> bytes:
            return bytes(xor_out)

        xr_string = input("input the text (string) to be encoded: ").encode("utf-8"
        xr_key = input("input the key (string) to be used to encode:  ").encode("ut
        result = xor_repeated(xr_string, xr_key)
        print(result.hex())
      ;e "0":
        break

      ;e  :
```

```
--------------------------------------------------------------------
Line 171 | key=53 ('5')
score=13.90  printable_ratio=1.00  alpha_ratio=0.80  spaces=5
plaintext:
Now that the party is jumping

--------------------------------------------------------------------
Line 103 | key=125 ('}')
score=12.89  printable_ratio=0.96  alpha_ratio=0.48  spaces=5
plaintext:
VD p | <N \XpC@?hPB(c9_X P
--------------------------------------------------------------------
Line 120 | key=17 ('?')
score=10.06  printable_ratio=0.85  alpha_ratio=0.26  spaces=4
plaintext:
\[?F -'  %2/XEHY fG-=
>
--------------------------------------------------------------------
Line 169 | key=119 ('w')
score=9.30  printable_ratio=0.92  alpha_ratio=0.64  spaces=3
plaintext:
DvG N#  Cq5]LztrJzZbkD@
--------------------------------------------------------------------
Line 306 | key=110 ('n')
```

5:

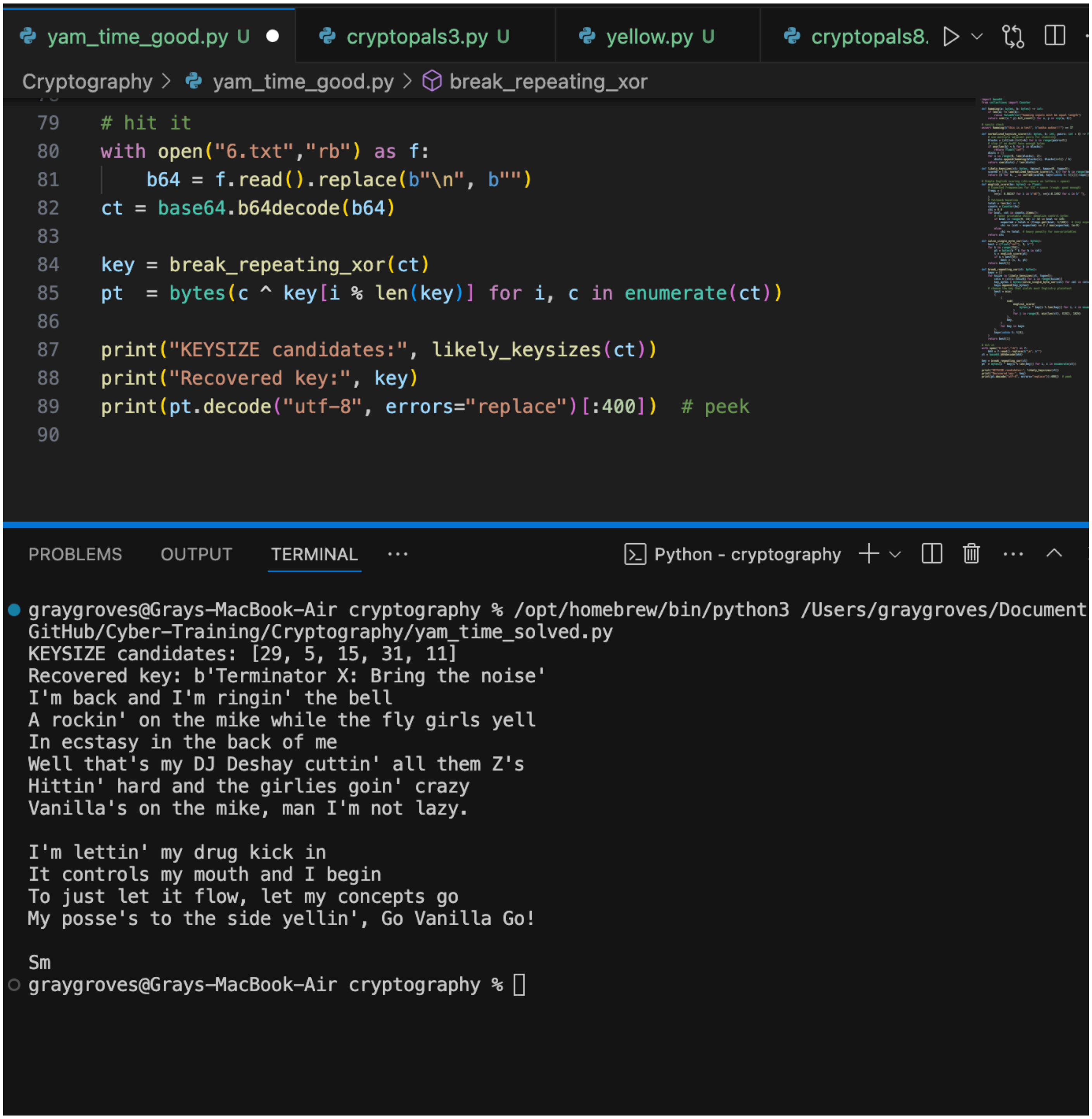


```
yam_time.py U    cryptopals3.py U    yellow.py U    cryptopals8.py U
Cryptography > cryptopals3.py > ...
            best_total, best_idx, best_key, best_pr, best_spaces, best_ar, be
            if (best_pr >= 0.90 and best_spaces >= 1) or (best_ar >= 0.55 and
                print(f"\n>>> PROBABLE single-byte XOR on line {best_idx}")

        case "4":
            def xor_repeated(string_in: bytes, key: bytes) -> bytes:
                xor_out = bytearray()
                for i in range(len(string_in)):
                    xor_out.append(string_in[i] ^ key[i % len(key)])
                return bytes(xor_out)

            print("paste plaintext, then send EOF (Ctrl-D twice on mac/linux,
            xr_string = sys.stdin.buffer.read()          # exact bytes, no
            xr_key = b"ICE"

            result = xor_repeated(xr_string, xr_key)
            print(result.hex())


            #Burning 'em, if you ain't quick and nimbleI go crazy when I hear
        case "0":
            break

        case _:
            print("Invalid option, select one of the provided options.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter a value: 4
Paste the TWO LINES of plaintext, then send EOF (Ctrl-D on mac/linux, Ctrl-Z then Enter on Windo
):
Burning 'em, if you ain't quick and nimble
I go crazy when I hear a cymbal0b3637272a2b2e63622c2e69692a23693a2a3c6324202d623d63343c2a2622632
72765272a282b2f20430a652e2c652a3124333a653e2b2027630c692b20283165286326302e27282f
Enter a value:
```

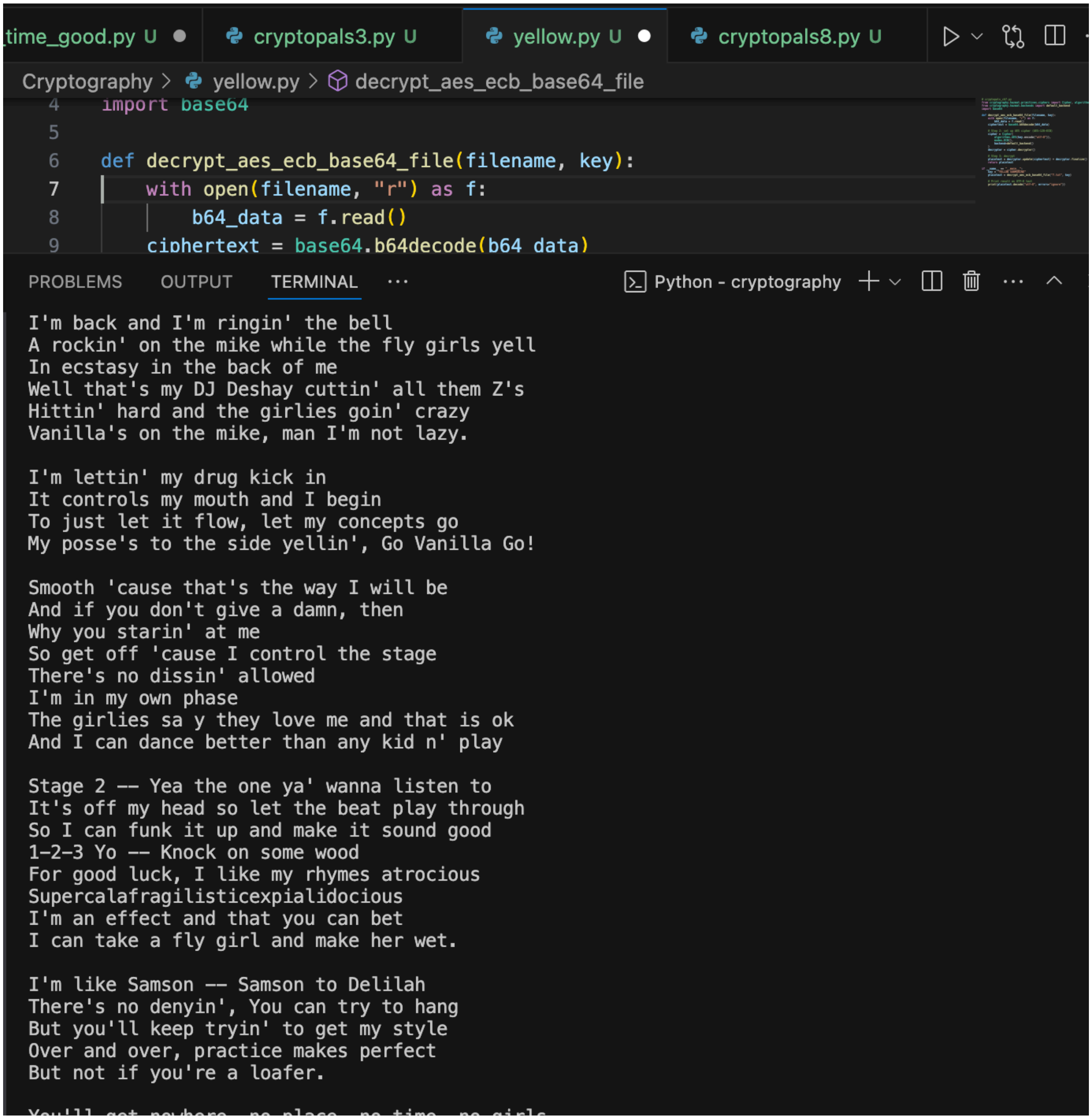
6:



```python
# hit it
with open("6.txt","rb") as f:
    b64 = f.read().replace(b"\n", b"")
ct = base64.b64decode(b64)

key = break_repeating_xor(ct)
pt  = bytes(c ^ key[i % len(key)] for i, c in enumerate(ct))

print("KEYSIZE candidates:", likely_keysizes(ct))
print("Recovered key:", key)
print(pt.decode("utf-8", errors="replace")[:400])  # peek
```

```
graygroves@Grays-MacBook-Air cryptography % /opt/homebrew/bin/python3 /Users/graygroves/Document
GitHub/Cyber-Training/Cryptography/yam_time_solved.py
KEYSIZE candidates: [29, 5, 15, 31, 11]
Recovered key: b'Terminator X: Bring the noise'
I'm back and I'm ringin' the bell
A rockin' on the mike while the fly girls yell
In ecstasy in the back of me
Well that's my DJ Deshay cuttin' all them Z's
Hittin' hard and the girlies goin' crazy
Vanilla's on the mike, man I'm not lazy.

I'm lettin' my drug kick in
It controls my mouth and I begin
To just let it flow, let my concepts go
My posse's to the side yellin', Go Vanilla Go!

Sm
graygroves@Grays-MacBook-Air cryptography % 
```

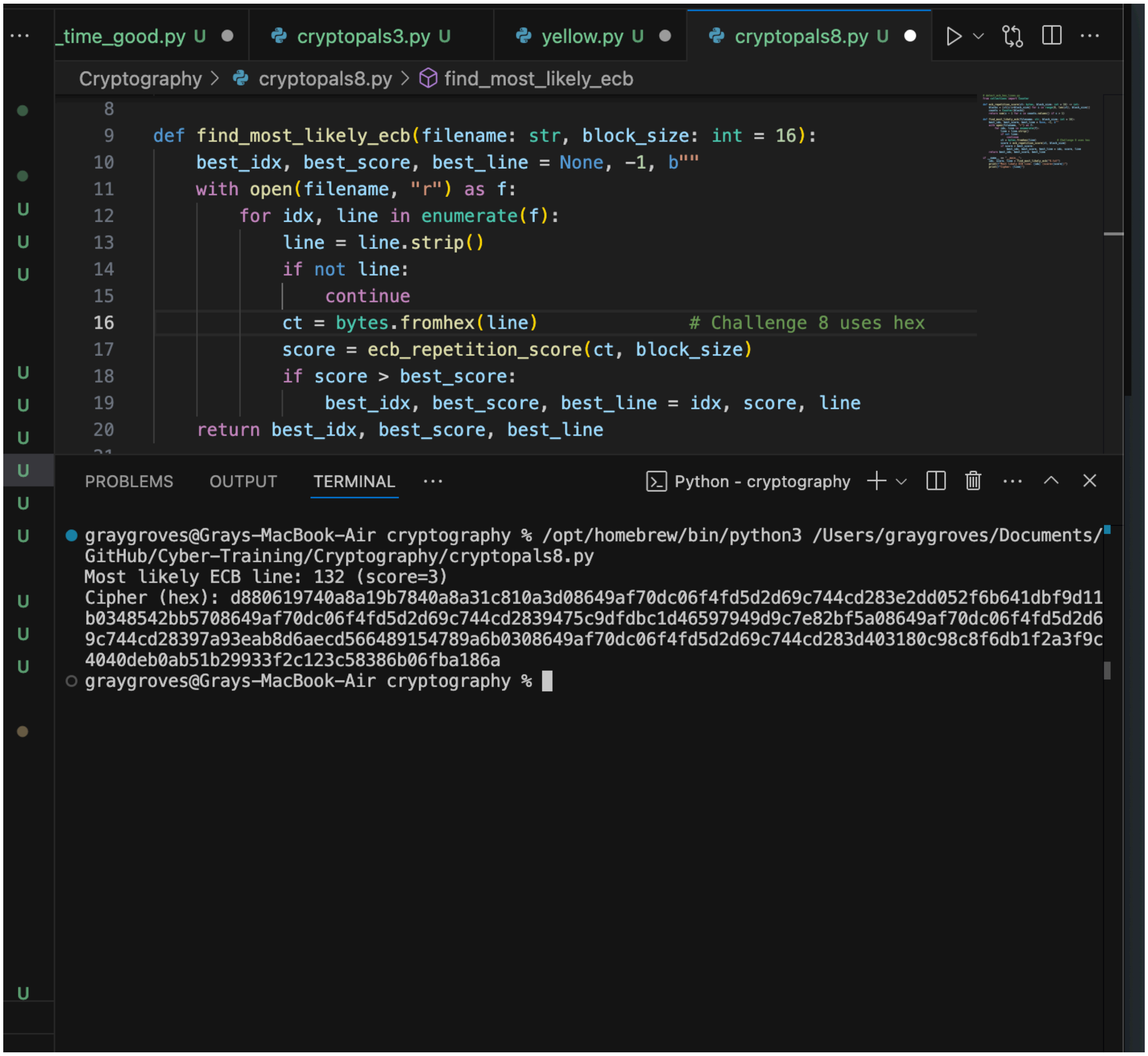
7:



```python
import base64

def decrypt_aes_ecb_base64_file(filename, key):
    with open(filename, "r") as f:
        b64_data = f.read()
    ciphertext = base64.b64decode(b64_data)
```

```
I'm back and I'm ringin' the bell
A rockin' on the mike while the fly girls yell
In ecstasy in the back of me
Well that's my DJ Deshay cuttin' all them Z's
Hittin' hard and the girlies goin' crazy
Vanilla's on the mike, man I'm not lazy.

I'm lettin' my drug kick in
It controls my mouth and I begin
To just let it flow, let my concepts go
My posse's to the side yellin', Go Vanilla Go!

Smooth 'cause that's the way I will be
And if you don't give a damn, then
Why you starin' at me
So get off 'cause I control the stage
There's no dissin' allowed
I'm in my own phase
The girlies sa y they love me and that is ok
And I can dance better than any kid n' play

Stage 2 -- Yea the one ya' wanna listen to
It's off my head so let the beat play through
So I can funk it up and make it sound good
1-2-3 Yo -- Knock on some wood
For good luck, I like my rhymes atrocious
Supercalafragilisticexpialidocious
I'm an effect and that you can bet
I can take a fly girl and make her wet.

I'm like Samson -- Samson to Delilah
There's no denyin', You can try to hang
But you'll keep tryin' to get my style
Over and over, practice makes perfect
But not if you're a loafer.

You'll get nowhere, no place, no time, no girls
```

8:



```python

def find_most_likely_ecb(filename: str, block_size: int = 16):
    best_idx, best_score, best_line = None, -1, b""
    with open(filename, "r") as f:
        for idx, line in enumerate(f):
            line = line.strip()
            if not line:
                continue
            ct = bytes.fromhex(line)              # Challenge 8 uses hex
            score = ecb_repetition_score(ct, block_size)
            if score > best_score:
                best_idx, best_score, best_line = idx, score, line
    return best_idx, best_score, best_line
```

```
graygroves@Grays-MacBook-Air cryptography % /opt/homebrew/bin/python3 /Users/graygroves/Documents/
GitHub/Cyber-Training/Cryptography/cryptopals8.py
Most likely ECB line: 132 (score=3)
Cipher (hex): d880619740a8a19b7840a8a31c810a3d08649af70dc06f4fd5d2d69c744cd283e2dd052f6b641dbf9d11
b0348542bb5708649af70dc06f4fd5d2d69c744cd2839475c9dfdbc1d46597949d9c7e82bf5a08649af70dc06f4fd5d2d6
9c744cd28397a93eab8d6aecd566489154789a6b0308649af70dc06f4fd5d2d69c744cd283d403180c98c8f6db1f2a3f9c
4040deb0ab51b29933f2c123c58386b06fba186a
graygroves@Grays-MacBook-Air cryptography %
```